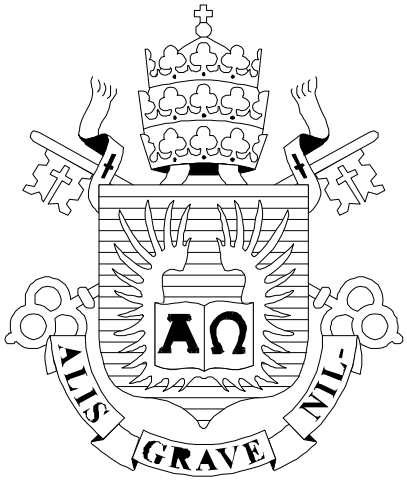

PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 20/08

Verificando e Otimizando Agentes Adaptativos Auto-Organizáveis

Maíra Athanázio de Cerqueira Gatti
Ruy Luis Milidiú
Carlos José Pereira de Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 20/08 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena April, 2008

Verificando e Otimizando Agentes Adaptativos Auto-Organizáveis

Maíra Athanázio de Cerqueira Gatti, Ruy Luiz Milidiú, Carlos José Pereira de Lucena

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brasil

{mgatti, milidiu, lucena}@inf.puc-rio.br

Abstract. We present in this paper the research done in order to use a learning technique to optimize the agents' self-organization. Those agents represent a biological system, more specifically, the stem cells behavior. We need to optimize their emergent behavior. To accomplish this goal, we used Plansim, a tool for planners' agents, and the LRTA* algorithm. We did some experiments in order to evaluate the research hypothesis's adequacy, and we described the work done, the results achieved with some discussion and the future work.

Keywords: Multi-agent Systems (MAS), Software Engineering for MAS, Adaptation, Verification, Online search, Planners.

Resumo. Neste artigo apresentamos a investigação realizada para a utilização de uma técnica de aprendizado para otimizar a auto-organização de agentes que representem um sistema biológico, no caso, células-tronco. Deseja-se otimizar seu comportamento de natureza emergente. Para isto, utilizamos o Plansim, um arcabouço para agentes planejadores, e o algoritmo A* com aprendizado em tempo real. Realizamos alguns experimentos para avaliar sua adequação e apresentamos o trabalho realizado, resultados obtidos e trabalhos futuros considerando tais resultados.

Palavras-chave: Sistemas Multiagentes, Engenharia de Software de Sistemas Multiagentes, Adaptação, Verificação, Busca online, Planejadores.

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br

I. Introdução

Células-tronco são células indiferenciadas capazes de se diferenciar em qualquer tipo de célula. Considerando a classificação de células-tronco existente, dependendo do seu grau de diferenciação, as células-tronco adultas, multipotentes, são as mais utilizadas por especialistas da área para evoluir a terapia *in vitro* de células-tronco. Uma das maiores preocupações destes cientistas é encontrar um modelo de diferenciação que satisfaça um conjunto de propriedades determinadas pela terapia. Eles precisam garantir, por exemplo, que ela não resultará em um tumor, ou que simplesmente será bem sucedida dada determinadas circunstâncias.

A geração de um modelo computacional para simular o comportamento de células-tronco é de fundamental importância para tais especialistas, visto que serve como uma terapia *in silico* que é muito mais barata que a terapia *in vitro,* realizada atualmente antes da *in vivo [1].*

Dado um simulador baseado em agentes que implemente tal modelo computacional [1], os especialistas desejam adicionar fatores de crescimento, de diferenciação e de morte celular para realizar a terapia que nada mais é que uma otimização deste comportamento emergente gerado pelas células-tronco. Do ponto de vista computacional, deseja-se otimizar o comportamento emergente dos agentes, isto é, deseja-se otimizar a auto-organização de sistemas multiagentes.

Vale ressaltar que, se por um lado já existe a complexidade do domínio de aplicação, por outro, otimizar tal auto-organização de sistemas multiagentes é um trabalho não trivial visto que se encontram muitos poucos trabalhos relacionados ao apoio do projeto de sistemas auto-organizáveis e menos ainda sobre otimização. Por isto a investigação da aplicação de um arcabouço para agentes planejadores, o Plansim, com a utilização de busca online através do algoritmos A* com aprendizado em tempo real.

II. Descrição do Problema

No modelo computacional de células-tronco baseado em agentes gerado, existem 4 tipos de células: **pluripotentes**, com elevado grau de diferenciação podendo gerar diferentes tipos de células, **precursora**, células capazes de se diferenciar em um determinado tipo de célula, por exemplo, sanguínea, **progenitora**, célula ainda mais comprometida com um tipo de célula e com a menor capacidade de diferenciação, e a **célula diferenciada** propriamente dita, sem nenhuma capacidade de diferenciação. Cada célula é representada por um agente. E a interação célula-célula e célula-ambiente é feita via interações entre os agentes e o ambiente.

Durante o processo de diferenciação uma célula pode ser dividir simetricamente, via mitose, gerando duas células-filha idênticas a si, ou assimetricamente, no caso de diferenciação, gerando uma célula-filha idêntica a si e uma parcial ou totalmente diferenciada.

De uma forma geral deseja-se otimizar a auto-organização de agentes adaptativos. De forma mais específica, e aplicada ao domínio de células-tronco, deseja-se:

- Minimizar morte celular: não é um processo trivial minimizar a morte celular simplesmente adicionando fatores de tal tipo, porque sabe-se empirica-

mente que ao se adicionar tais fatores na cultura, aumenta o número de células diferenciadas não desejadas;

- Maximizar número total de células diferenciadas: existem diversos fatores internos da colônia que influenciam tal processo de diferenciação;
- Maximizar o número de um determinado tipo de célula diferenciada: deseja-se que somente ou a grande maioria de células a se diferenciarem sejam células neurais, mais especificamente, neurônio.
- Minimizar probabilidade de tumor: além da proliferação desordenada da células-tronco como uma possível geração de tumor, células diferenciadas com mais cromossomos viram tumor se não morrerem.

Deseja-se encontrar uma taxa de crescimento ótimo onde todas estas condições sejam satisfeitas.

III. Planejadores de Busca Online e LRTA*

Em um problema não determinístico, estocástico e dinâmico, onde não se conhece todos os estados alcançáveis, utilizar estratégias de busca heurística *offline* (planejamento clássico) é insuficiente e inadequado. Para isto é necessário utilizar estratégias de busca *online*, onde o planejamento é intercalado com a ação. Isto é, a cada vez que uma ação é executada, observa-se o ambiente para calcular a próxima ação.

Um agente de busca online conhece as ações possíveis para um dado estado e sabe se esta ação pertence ao seu objetivo atual. Porém não sabe o estado sucessor uma vez tomada esta ação e nem o custo desta ação. Desta forma, à medida em que vai executando as ações, ele memoriza as ações já realizadas e atualiza o seu espaço de busca. Se o agente percebe que não está atingindo o objetivo, ela retorna a estados anteriores e executa novas ações. Assim, as ações devem ser reversíveis.

Uma forma de dar mais memória para o agente e de armazenar a melhor estimativa de custo para ser atingir um objetivo a partir de uma ação é a utilização do algoritmo A* com aprendizado em tempo real (LRTA* - Learning Real-Time A*) [3].

Desta forma, utilizamos o Plansim [2], um arcabouço de software para o desenvolvimento de planejadores especializados desenvolvido em C++. Ele define uma arquitetura para planejadores que utilizem busca heurística direta como mecanismo de planejamento e simuladores de eventos discretos para o modelo do sistema, fixando as interfaces de componentes como objetivos, funções heurísticas e simuladores. Além disso, provê diversas estratégias de busca, inclusive o algoritmo LRTA*.

IV. Trabalho Realizado

Como mencionado, o Plansim fixa interfaces de componentes como objetivos e simuladores, possuindo a seguinte estrutura:

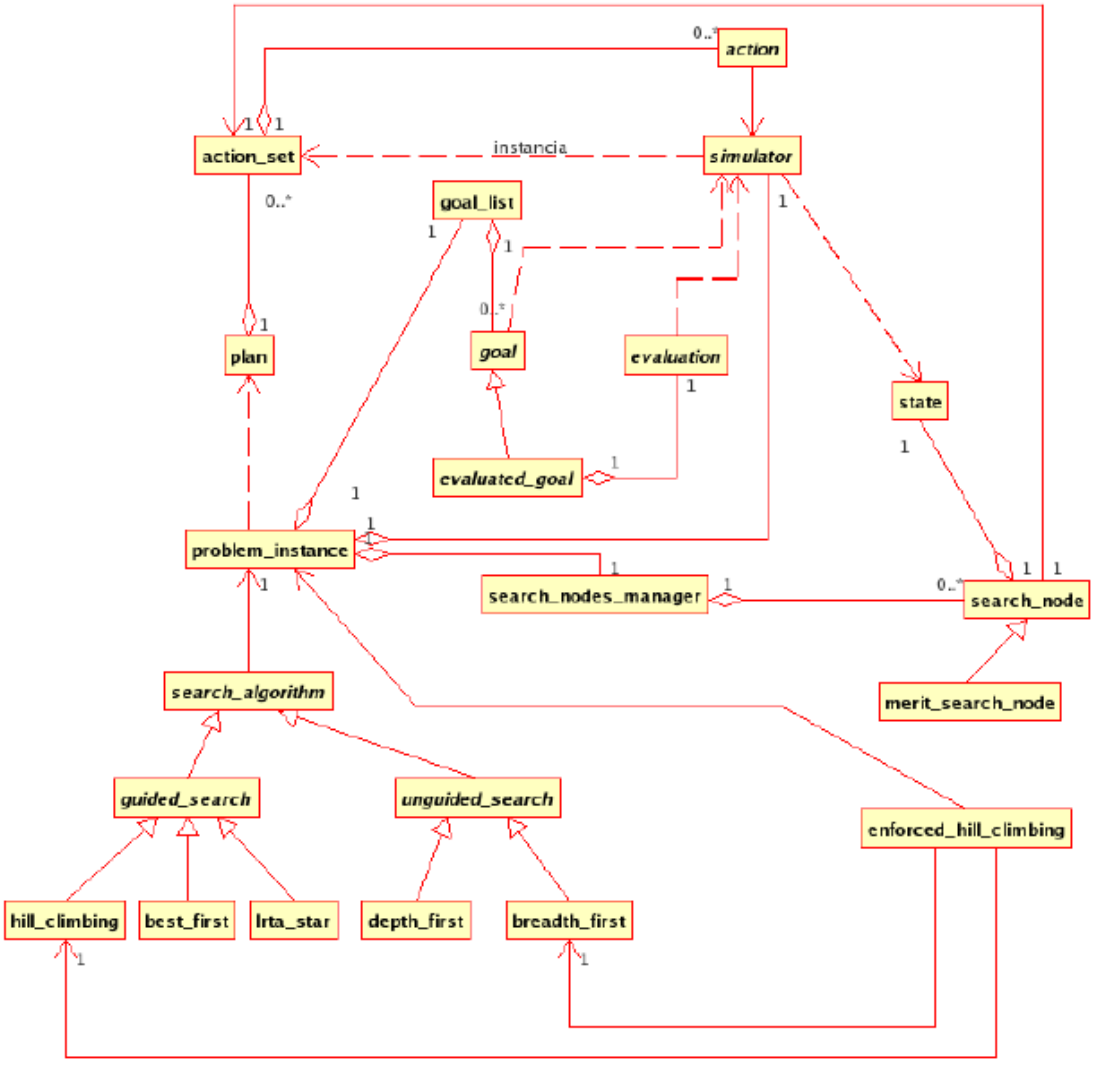

Figura 1: Estrutura do Plansim [2]

As classes derivadas de search_*agorithm* e a classe *lrts_star* são responsáveis pela execução das estratégias de busca do PLANSIM. As classes *search_nodes_manager*, *search_node* e *merit_search_node* são finais e responsáveis pelo gerenciamento do espaço de estados explorado. As classes *plan*, *action*, *action_set*, *goal* e *evaluation* representam os conceitos clássicos da área de planejamento, com *evaluation* representando as funções heurísticas. As classes *goal* e *evaluation* são hot spots e foram instanciadas como será visto posteriormente. As classes *simulator* e *state* definem a interface necessária para os simuladores e a representação dos seus estados. A classe *simulator* também precisou ser instanciada. Finalmente a classe *problem_instance* é responsável pela representação de uma instância de um problema de planejamento, contendo referências para o simulador utilizado, os objetivos e o gerenciador de estados explorados. Isto é, onde criamos as inicializações destes componentes e do simulador em si do comportamento das células-tronco.

Solução Proposta

O primeiro passo antes da instanciação do PLANSIM foi criar o algoritmo de decisão que exibe todas as ações possíveis do simulador, e adicionar as ações simétricas para o caso de desfazer um caminho em busca do caminho ótimo no espaço de estados.

A figura 2 ilustra este algoritmo. De uma forma geral, após o simulador ser inicializado com o número inicial de células-tronco desejado pelo especialista, existem dois tipos de ações que podem ocorrer: externas e internas. As ações externas são adicionar ou remover fatores de proliferação e diferenciação.

Neste primeiro momento não foi incluída a ação de adicionar ou remover fator de morte celular por questões de tempo de desenvolvimento. Entretanto, seria uma atividade bastante trivial que seguiria praticamente o mesmo protocolo que as outras.

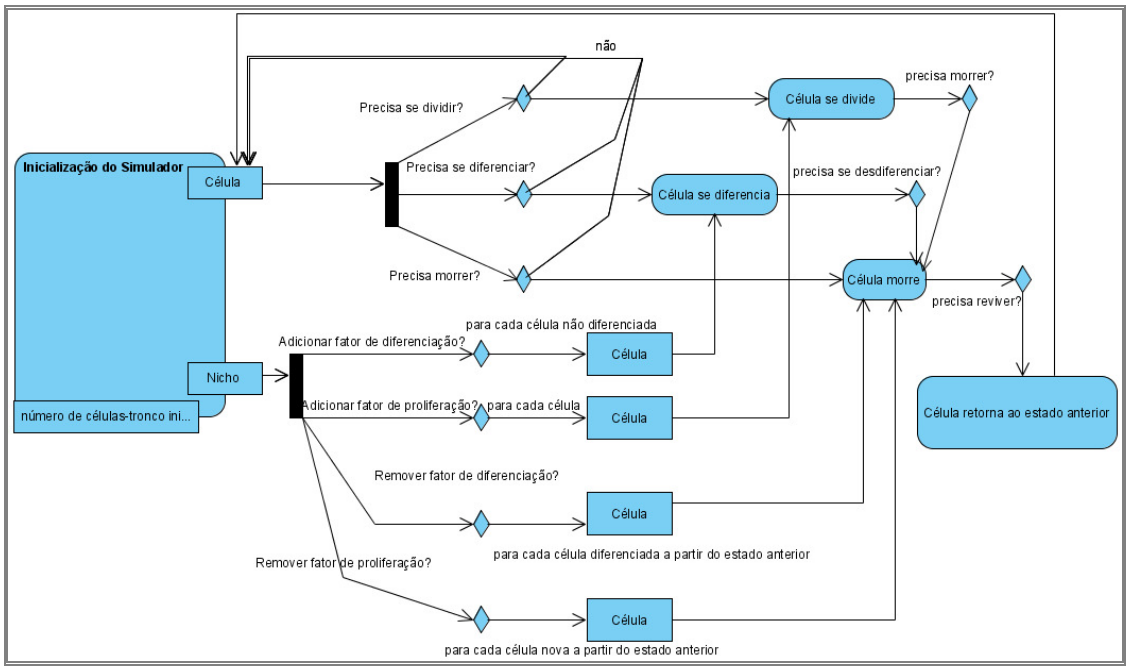

Figura 2 - Algoritmo de Decisão

As ações internas são relativas ao estado das células e são tomadas de forma indiferente: ou via processos intra-celulares, ou via ações externas. Dado que o Nicho é o ambiente regulador no qual as células vivem, ele foi representado como o simulador que recebe as ações externas.

Para facilitar a visualização do algoritmo, a maioria dos fluxos de controle que saem dos nós de decisão com não, foram omitidos. Porém eles retornariam ao objeto de controle em questão: célula ou nicho.

Instanciação

Uma vez definido o algoritmo e o framework PLANSIM, sua instanciação foi quase que automática. O simulador da aplicação, a classe **niche**, estende da classe *simulator*. As ações externas que modificam o estado do simulador diretamente estendem da classe *action* e as classes que modificam o estado do simulador indiretamente alterando os estados das células e sua quantidade no simulador estendem da classe **cell_action** que por sua vez estende da classe *action* e modifica o estado da célula (figura 3).

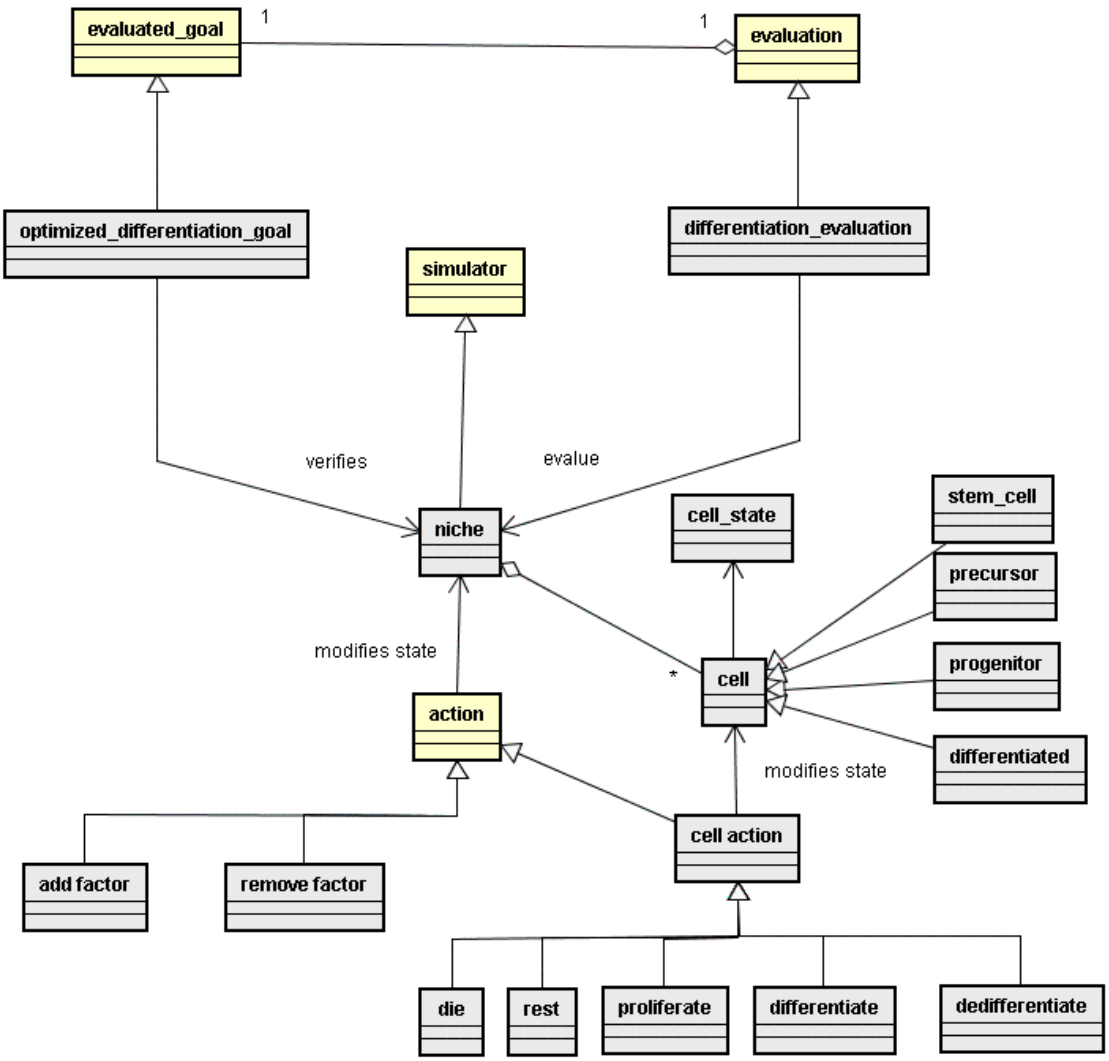

Figura 3 - Instanciação do PLANSIM

A classe **optimized_differentiation_goal** define a função heurística para avaliação do estado e estende da classe *evaluated_goal*. A classe **differentiation_evaluation** retorna verdade para um subconjunto de estados, definido pelo número de células, e estende a classe *evaluation*.

Para definir o estado ótimo, ou neste caso o estado aceitável, foram definidos alguns *thresholds* em relação ao número de cada tipo de célula em seu respectivo estado, isto é, proliferando, morta, se diferenciando, etc (para cada uma das ações definidas), como segue:

- O **número de células multipotentes**, isto é, com capacidade de 100% de diferenciação deve ser **maior ou igual** ao número de células tronco inicial **e menor ou igual** a duas vezes o número de células tronco inicial; **&**
- O **número de células precursoras**, isto é, com capacidade de 50% de diferenciação, deve ser **maior ou igual** ao número de células multipotentes **e menor ou igual** a duas vezes o número de células multipotentes; **&**
- O **número de células progenitoras**, isto é, com capacidade de 25% de diferenciação, deve ser maior ou igual número de células precursoras e menor ou igual a duas vezes o número de células precursoras; **&**

- O **número de células diferenciadas** deve ser igual ao número de células progenitoras; **&**, finalmente,
- O **número de células mortas** deve ser sempre menor ou igual a metade do número de células multipotentes, e/ ou menor ou igual a metade do número de células precursoras se e somente se existirem células precursoras, e/ ou menor ou igual a metade do número de células progenitoras, se existirem também.

Dificuldades encontradas

Primeiramente, é importante ressaltar que a definição deste modelo foi incremental. Primeiro tentou-se gerar uma versão da instanciação sem considerar as ações externas como ações. Neste momento, além de ter descoberto este problema, percebeu-se o quão complexo seria para paralelizar as ações que, para gerar um modelo real, devem ser paralelizadas. Seria necessário o uso de MPI (Message Processing Interface) [4] para definir cada ação como um processo.

Para o desenvolvimento de tal tarefa seria necessário distribuir os agentes em um grid para que as ações fossem distribuídas e ganhar em escalabilidade, já que é necessário trabalho com um número muito elevado de células até obter o comportamento observável desejado. Como esta seria uma tarefa árdua e não seria realizada no tempo disponível do trabalho, foi uma decisão de projeto projetar e implementar como uma simulação serial somente para prova de conceito da aplicabilidade do Plansim neste problema de domínio.

Depois de um grande esforço em configurar o Plansim, que foi desenvolvido e testado no Linux, para o ambiente Windows, o desenvolvimento da instanciação se deu de forma rápida. Entretanto, chegou-se a uma decisão de projeto no qual conclui-se que o mais adequado seria evoluir a proposta do acabouço Plansim do que adaptar o modelo do comportamento ao mesmo, visto que não só tornaria o modelo mais complexo, como diferente do comportamento real. De forma geral , o maior problema se deu porque o Plansim considera o simulador de eventos discretos como uma caixa preta, isto é, objetos contidos nele cujo espaço de estados precisam ser tratados na função heurística não são considerados, somente o simulador em si. Desta forma, somente o **niche** era avaliado pela função e no momento em que era preciso reverter as ações, caso o caminho percorrido fosse em uma direção não desejada do objetivo, o arcabouço se mostrou incapaz de reverter as ações das células, representadas em um nível mais abstrato pela classe **cell**.

V. Conclusões e Trabalhos Futuros

Este documento apresentou o trabalho realizado na tentativa de otimizar o comportamento emergente gerado pela auto-organização de células-tronco representadas por agentes. O objetivo era definir propriedades macros e a partir de interações locais e a utilização de um planejador especializado de busca online que utiliza o algoritmo A* com aprendizado em tempo real otimizar o comportamento de forma que as propriedades macros sejam satisfeitas.

A principal contribuição deste trabalho foi a demonstração de como utilizar uma ferramenta de verificação capaz de apoiar tanto a fase de especificação de propriedades macros de um sistema multiagentes auto-organizável, quanto de verificar esta auto-organização a partir de tais propriedades via simulação.

Como trabalho futuro, pretende-se re-projetar o acarbouço Plansim em Java, evoluindo o seu projeto para permitir uma recursão na busca pelo estado ótimo a partir de um simulador composto de várias entidades e integrando ao framework de simulação de sistemas multiagentes, Mason [5], que já disponibiliza diversos componentes para a definição de um simulador de eventos discretos baseado em agentes e com definição de ações síncronas, assíncronas e compostas, assim como permite visualização 2D e 3D da simulação.

Bibliografia

[1] Gatti, M.; de Vasconcelos, J.E.; Lucena, C.; An Agent Oriented Software Engineering Approach for the Adult Stem-Cell Modeling, Simulation and Visualization. In: Third Workshop on Software Engineering for Agent-oriented Systems - SEAS, 2007, João Pessoa. Software Engineering for Agent-oriented Systems, 2007.

[2] Liporace, F. dos S.; Milidiú, R. L.; Planejadores para transporte em polidutos. Rio de Janeiro, 2005. 120 pg. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[3] Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence. G. Weiss (ed). MIT Press, 2001. Chapter 4, pp. 165-199.

[4] Gropp, W., Lusk, E., Skjellum, A., Using MPI: Portable Parallel Programming with the Message Passing-Interface. MIT Press, 1994.

[5] Luke, S., Cioffi-Revilla, C., Panait, L. and Sullivan, K. (2004), 'MASON: A New Multi-Agent Simulation Toolkit', SwarmFest 2004, Eighth Annual Swarm Users/Researchers Conference., University of Michigan, Ann Arbor, Michigan USA.